

Московский государственный технический университет им. Н.Э. Баумана  
Кафедра «Системы обработки информации и управления»



Лабораторная работа №2  
по дисциплине  
«Методы машинного обучения»  
на тему

**«Разведочный анализ данных. Исследование и визуализация данных»**

Выполнил:  
студент группы ИУ5И-21М  
Ван Чжэн

Москва — 2024 г.

# 1.Цель лабораторной работы:

Изучение продвинутых способов предварительной обработки данных для дальнейшего формирования моделей.

## 2.Задание:

1. Выбрать набор данных (датасет), содержащий категориальные и числовые признаки и пропуски в данных. Для выполнения следующих пунктов можно использовать несколько различных наборов данных (один для обработки пропусков, другой для категориальных признаков и т.д.)
2. Для выбранного датасета (датасетов) на основе материалов лекций решить следующие задачи:
  - устранение пропусков в данных;
  - кодирование категориальных признаков;
  - нормализацию числовых признаков.

## 3.Ход выполнения работы

Набор данных-«Нью-Йорк - Почасовые данные о погоде».

Все данные - 10481\*30.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import sklearn.impute
import sklearn.preprocessing

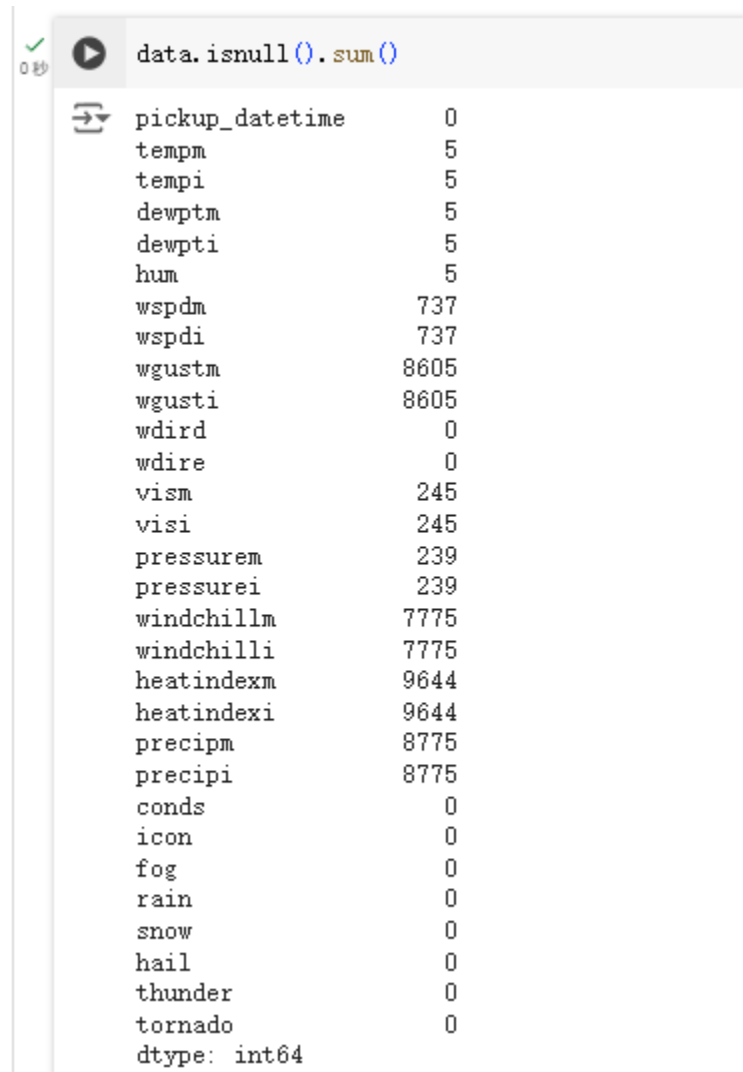
data = pd.read_csv('/content/Weather.csv')
data
```

	pickup_datetime	tempm	tempf	dewptm	dewptf	hum	wspd	wspdi	wgustm	wgustf	...	precipm	precipf	conds	icon	fog	rain	snow	hail	thunder	tornado
0	2015-12-31 00:15:00	7.8	46.0	6.1	43.0	89.0	7.4	4.6	NaN	NaN	...	0.5	0.02	Light Rain	rain	0	1	0	0	0	0
1	2015-12-31 00:42:00	7.8	46.0	6.1	43.0	89.0	7.4	4.6	NaN	NaN	...	0.8	0.03	Overcast	cloudy	0	0	0	0	0	0
2	2015-12-31 00:51:00	7.8	46.0	6.1	43.0	89.0	5.6	3.5	NaN	NaN	...	0.8	0.03	Overcast	cloudy	0	0	0	0	0	0
3	2015-12-31 01:51:00	7.2	45.0	5.6	42.1	90.0	7.4	4.6	NaN	NaN	...	0.3	0.01	Overcast	cloudy	0	0	0	0	0	0
4	2015-12-31 02:51:00	7.2	45.0	5.6	42.1	90.0	0.0	0.0	NaN	NaN	...	NaN	NaN	Overcast	cloudy	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
10476	2016-12-31 19:51:00	6.1	43.0	-4.4	24.1	47.0	7.4	4.6	NaN	NaN	...	NaN	NaN	Overcast	cloudy	0	0	0	0	0	0
10477	2016-12-31 20:51:00	6.1	43.0	-4.4	24.1	47.0	13.0	8.1	38.9	24.2	...	NaN	NaN	Overcast	cloudy	0	0	0	0	0	0
10478	2016-12-31 21:51:00	6.1	43.0	-5.0	23.0	45.0	9.3	5.8	29.6	18.4	...	NaN	NaN	Overcast	cloudy	0	0	0	0	0	0
10479	2016-12-31 22:51:00	6.7	44.1	-5.0	23.0	43.0	14.8	9.2	NaN	NaN	...	NaN	NaN	Overcast	cloudy	0	0	0	0	0	0
10480	2016-12-31 23:51:00	6.7	44.1	-4.4	24.1	45.0	7.4	4.6	25.9	16.1	...	NaN	NaN	Overcast	cloudy	0	0	0	0	0	0

10481 rows x 30 columns

### 3.1 Обработка пропусков в данных

Найдем все пропуски в данных:



The screenshot shows a Jupyter Notebook cell with the command `data.isnull().sum()` and its output. The output is a series of 31 variables and their corresponding count of null values. The variables are listed on the left, and the counts are on the right. The data type for the counts is `dtype: int64`.

pickup_datetime	0
tempm	5
tempi	5
dewptm	5
dewpti	5
hum	5
wspdm	737
wspdi	737
wgustm	8605
wgusti	8605
wdird	0
wdire	0
vism	245
visi	245
pressurem	239
pressurei	239
windchillm	7775
windchilli	7775
heatindexm	9644
heatindexi	9644
precipm	8775
precipi	8775
conds	0
icon	0
fog	0
rain	0
snow	0
hail	0
thunder	0
tornado	0
dtype: int64	

Очевидно, что мы можем выбрать столбец «vism»( Vivibility in Km/Видимость в км). Давайте попробуем четыре разные стратегии заполнения на основе: «среднее», «медиана», «наиболее часто встречающаяся» и «постоянная»

✓  
0秒

```
# Пример работы MissingIndicator
temp_x1 = np.array([[np.nan, 1, 3], [np.nan, 0, 5], [3, np.nan, 1]])
print('Исходный массив:')
print(temp_x1)
indicator = MissingIndicator(features='all')
temp_x1_transformed = indicator.fit_transform(temp_x1)
print('Маска пропущенных значений:')
print(temp_x1_transformed)
```

⇒ Исходный массив:

```
[[nan 1. 3.]
 [nan 0. 5.]
 [ 3. nan 1.]]
```

Маска пропущенных значений:

```
[[ True False False]
 [ True False False]
 [False  True False]]
```

✓  
0秒

```
def impute_column(dataset, column, strategy_param, fill_value_param=None):
    """
    Заполнение пропусков в одном признаке
    """
    temp_data = dataset[[column]].values
    size = temp_data.shape[0]

    indicator = MissingIndicator()
    mask_missing_values_only = indicator.fit_transform(temp_data)

    imputer = SimpleImputer(strategy=strategy_param,
                           fill_value=fill_value_param)
    all_data = imputer.fit_transform(temp_data)

    missed_data = temp_data[mask_missing_values_only]
    filled_data = all_data[mask_missing_values_only]

    return all_data.reshape((size,)), filled_data, missed_data
```

✓  
0秒

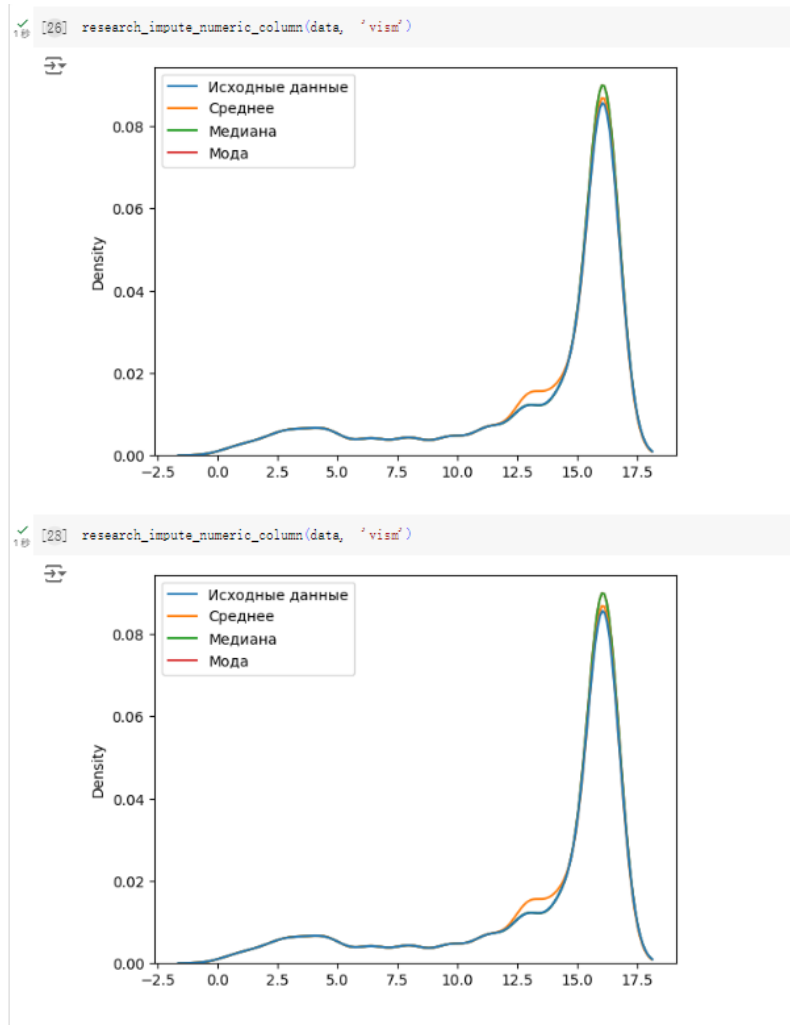
```
[24] def research_impute_numeric_column(dataset, num_column, const_value=None):
    strategy_params = ['mean', 'median', 'most_frequent', 'constant']
    strategy_params_names = ['Среднее', 'Медиана', 'Мода']
    strategy_params_names.append('Константа = ' + str(const_value))

    original_temp_data = dataset[[num_column]].values
    size = original_temp_data.shape[0]
    original_data = original_temp_data.reshape((size,))

    new_df = pd.DataFrame({'Исходные данные': original_data})

    for i in range(len(strategy_params)):
        strategy = strategy_params[i]
        col_name = strategy_params_names[i]
        if (strategy != 'constant') or (strategy == 'constant' and const_value != None):
            if strategy == 'constant':
                temp_data, _, _ = impute_column(dataset, num_column, strategy, fill_value_param=const_value)
            else:
                temp_data, _, _ = impute_column(dataset, num_column, strategy)
            new_df[col_name] = temp_data

    sns.kdeplot(data=new_df)
```



Распределения одномодальные, поэтому можно использовать для импутации моды.

### 3.2 Кодирование категориальных признаков

Я выбрал ещё набор данных-«продажа поддержанных автомобилей». Я использовал One-hot encoding для кодирования категориальных признаков.

```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
from sklearn.impute import MissingIndicator
from sklearn.impute import KNNImputer
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Lasso
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from IPython.display import Image

import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

data = pd.read_csv('/content/second_hand_car_sales.csv')
data
```

	Manufacturer	Model	Engine Size (L)	Fuel Type	Year of Manufacture	Mileage	Price
0	Mercedes-Benz	Cruze	1.418475	Electric	2013	61837	34792
1	Toyota	A4	4.492330	Electric	2003	128993	27129
2	Audi	C-Class	4.739375	Electric	2000	81362	29141
3	Nissan	Model 3	3.128423	Petrol	2011	168204	24731
4	Mercedes-Benz	Golf	1.650279	Diesel	2006	119405	27493
...	...	...	...	...	...	...	...
49995	Chevrolet	Corolla	1.241130	Diesel	2021	163295	1110
49996	Nissan	Civic	3.741902	Electric	2012	85805	27877
49997	Toyota	Altima	2.501539	Hybrid	2016	187733	42132
49998	Ford	Model 3	2.066934	Electric	2022	136728	39121
49999	Chevrolet	Fiesta	3.261779	Petrol	2000	104033	42007

50000 rows x 7 columns

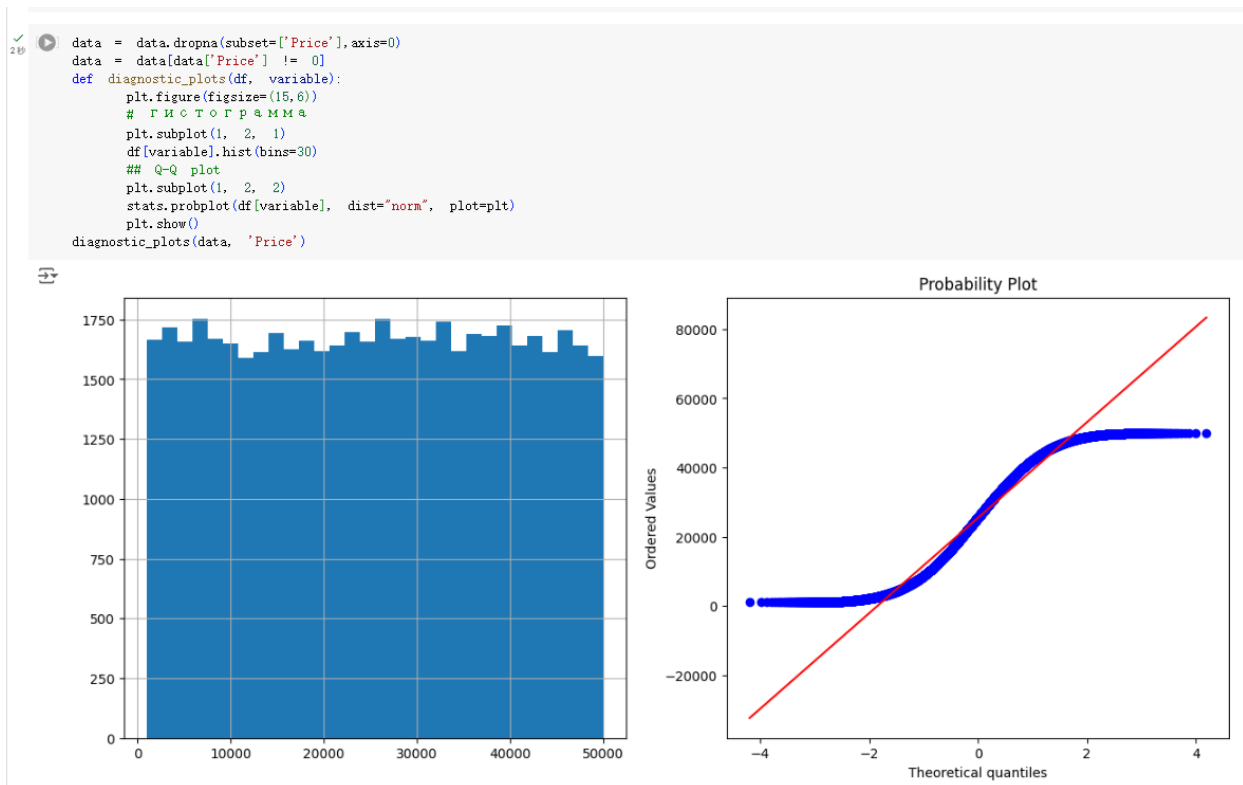
后续步骤: [查看推荐的图表](#)

```
from category_encoders.one_hot import OneHotEncoder as ce_OneHotEncoder
ce_OneHotEncoder1 = ce_OneHotEncoder()
data_OHE = ce_OneHotEncoder1.fit_transform(data[data.columns.difference(['Fuel', 'Type'])])
data_OHE
```

	Engine Size (L)	Manufacturer_1	Manufacturer_2	Manufacturer_3	Manufacturer_4	Manufacturer_5	Manufacturer_6	Manufacturer_7	Manufacturer_8	Manufacturer_9	...	Model
0	1.418475	1	0	0	0	0	0	0	0	0	0	...
1	4.492330	0	1	0	0	0	0	0	0	0	0	...
2	4.739375	0	0	1	0	0	0	0	0	0	0	...
3	3.128423	0	0	0	1	0	0	0	0	0	0	...
4	1.650279	1	0	0	0	0	0	0	0	0	0	...
...	...	...	...	...	...	...	...	...	...	...	...	...
49995	1.241130	0	0	0	0	0	1	0	0	0	0	...
49996	3.741902	0	0	0	1	0	0	0	0	0	0	...
49997	2.501539	0	1	0	0	0	0	0	0	0	0	...
49998	2.066934	0	0	0	0	0	0	0	0	0	0	...
49999	3.261779	0	0	0	0	0	1	0	0	0	0	...

50000 rows x 24 columns

### 3.3 Нормализация числовых признаков



### Список литературы

- [1] Гапанюк Ю. Е. Лабораторная работа «Обработка пропусков в данных, кодирование категориальных признаков, масштабирование данных» [Электронный ресурс] // GitHub. — 2019. — Режим доступа: [https://github.com/ugapanyuk/ml\\_course/wiki/LAB\\_MISSING](https://github.com/ugapanyuk/ml_course/wiki/LAB_MISSING) (дата обращения: 05.04.2019).
- [2] Team The IPython Development. IPython 7.3.0 Documentation [Electronic resource]//Read the Docs. — 2019. — Access mode: <https://ipython.readthedocs.io/en/stable/> (online; accessed: 20.02.2019).
- [3] Waskom M. seaborn 0.9.0 documentation [Electronic resource] // PyData. — 2018. — Access mode: <https://seaborn.pydata.org/> (online; accessed: 20.02.2019).
- [4] pandas 0.24.1 documentation [Electronic resource] // PyData. — 2019. — Access mode: <http://pandas.pydata.org/pandas-docs/stable/> (online; accessed: 20.02.2019).
- [5] Gupta L. Google Play Store Apps [Electronic resource] // Kaggle. — 2019. — Access mode: <https://www.kaggle.com/lava18/google-play-store-apps> (online; accessed: 05.04.2019).